Complex aggregates over clusters of elements

Celine Vens[1,2,3], Sofie Van Gassen[4], Tom Dhaene[4], and Yvan Saeys[1,2]

[1] Department of Respiratory Medicine, Ghent University, Ghent, Belgium
[2] VIB Inflammation Research Center, Ghent, Belgium
[3] Department of Public Health and Primary Care, KU Leuven Kulak, Kortrijk, Belgium
[4] Department of Information Technology (INTEC)-iMinds, Ghent University, Ghent, Belgium
Celine.Vens@kuleuven-kulak.be
{Sofie.VanGassen,Tom.Dhaene}@intec.ugent.be
Yvan.Saeys@irc.vib-ugent.be

Abstract. Complex aggregates have been proposed as a way to bridge the gap between approaches that handle sets by imposing conditions on specific elements, and approaches that handle them by imposing conditions on aggregated values. A complex aggregate summarises a subset of the elements in a set, where this subset is defined by conditions on the attribute values. In this paper, we present a new type of complex aggregate, where this subset is defined to be a cluster of the set. This is useful if subsets that are relevant for the task at hand are difficult to describe in terms of attribute conditions. This work is motivated from the analysis of flow cytometry data, where the sets are cells, and the subsets are cell populations. We describe two approaches to aggregate over clusters on an abstract level, and validate one of them empirically, motivating future research in this direction.

Keywords: relational learning, aggregation, clustering, flow cytometry

1 Introduction

In relational learning, examples are described by data residing in multiple tables of a relational database, linked together via one-to-one, one-to-many or many-to-many relationships. The latter two types of relationships result in single examples being related to a *set* of objects, and require mechanisms to deal with such sets. While, traditionally, relational learning approaches handled sets by either looking at properties of individual elements or by aggregating over them, a number of researchers [12, 14, 17, 20, 21] have looked into the combination of these approaches, resulting in so-called complex aggregates. Such complex aggregates aggregate over a subset of the elements of a set, where the subset is usually defined by imposing conditions on some attribute values. For instance, consider the widely used Mutagenesis dataset [18] of molecules that are classified as mutagenic or not, and are described by their atoms and bonds. Thus, any molecule is related to a *set* of atoms and bonds. A complex aggregate in this

context is the number of carbon atoms of a molecule: it consists of an aggregation (namely, count) over a subset of the atoms (namely, the carbon atoms). Another example is the maximum charge of the atoms that are bound to a carbon atom. While for this application it makes sense to describe subsets of atoms by imposing conditions on their element or on the atoms they are bound to, for other applications it is less obvious how to describe useful subsets based on attribute conditions or on related objects. In flow cytometry data analysis for instance, one deals with patient samples (e.g., blood samples), which are described by the individual cells they contain. Each cell in turn is described by some 20 numeric fluorescence intensities. For certain auto-immunity diseases, it is known that the number of cells of a particular cell population (e.g., B-cells, eosinophils,...) is affected. These cell populations are difficult to describe in terms of conditions on the intensity markers, however. They are usually detected by a (manual) clustering procedure over the set of cells. Thus, for such applications, it makes sense to aggregate over subsets defined by clusters instead of defined by attribute conditions.

The contributions of this paper are the following: (1) We present a new type of complex aggregate: aggregates over clusters; (2) We motivate this from a real world example, namely flow cytometry data analysis; (3) We present a first approach to use aggregates over clusters; (4) By showing promising results on both a synthetic and a real world dataset, we motivate future research in this direction.

2 Related work

Traditionally, relational learners typically handled sets in one of two approaches. The first approach handled sets by checking properties of specific elements. Most inductive logic programming (ILP) [15] systems use this approach, using the existential quantifier to check individual elements. The second approach handled sets by aggregating over them, reducing a complete set to one or a few values (e.g., number of elements in the set, average value of one of the attributes of the set,...). Examples of this approach include probabilistic relational models [13] and relational probability trees [16]. Blockeel and Bruynooghe [5] discussed the resulting - often undesirable - bias on these learners, and proposed the idea of combining aggregation and selection. Challenges in this combination are the large number of features that may be generated and the fact that it is more difficult to traverse the feature space in a structured and efficient way.

Krogel and Wrobel [14] introduced aggregate functions that apply not only to single attributes, but also to pairs of attributes, one of which has to be nominal and serves as a *group by* condition. The resulting aggregate conditions are still of limited complexity and are not refined further during the search. Knobbe et al. [12] proposed a method for subsequently specializing the set to be aggregated. By restricting the application of this specialization operator to aggregate functions where its effect is well-understood, they can search the hypothesis space in a general-to-specific way, but this obviously limits the kind of complex conditions

that can be found. Uwents and Blockeel [20] described relational neural networks as a subsymbolic approach towards learning complex aggregates. Their approach is not constrained to using predefined aggregate functions and does not make a distinction between searching for aggregate functions and searching for complex conditions, but the resulting theories are also not interpretable in terms of well-understood aggregates and conditions.

Van Assche et al. [21] have made the first implementation of combined aggregates and selections in an ILP system. They have extended the refinement operator of the relational decision tree learner TILDE [3] to include so-called complex aggregates: literals of the form $F(V, Q, R)$, where F is an aggregate function (e.g., count), V is an aggregate variable occurring in the aggregate query Q, and R is the result of applying F to the set of all answer substitutions for V that Q results in (we will call this set the result set of Q). A complex aggregate is usually followed by a condition on this result set. The "complex" part of a complex aggregate refers to Q being arbitrarily complex. A complex aggregate can be constructed by iterative refinement of Q, starting with a general query (e.g., the number of atoms of a molecule) and ending with a very specific one (e.g., the number of carbon atoms bound with an aromatic bond type to an atom with charge larger than 0.06). The feature explosion resulting from combining aggregate functions with selection conditions was handled by upgrading TILDE to a random forest [21] and taking advantage of the feature sampling applied at each node of the trees. Charnay et al. [6] have recently proposed an alternative solution, by introducing a hill-climbing approach to build complex aggregates incrementally.

Other types of complex aggregates that have been proposed include count-of-count features, which are a kind of nested aggregate, and have been introduced by [9, 11] in the type extension tree representation language. An example of such a feature is the complete specification of how many atoms that are present in the molecule to be classified are bound to how many atoms with an aromatic bond type.

Frank et al. [8] have introduced multi-feature aggregation functions in relational learning. In this setting the aggregate variable V is replaced by a set of variables, and this allows the use of aggregate functions like correlation or slope of line of best fit.

3 Case study: Flow cytrometry data analysis

Flow cytometry [1, 10] allows to quantify different cell populations in large numbers of cells, by suspending the cells in a stream of fluid and passing them through a laser beam, while measuring the resulting fluorescent and scattered light. Flow cytometry is applied both in clinical settings (in the diagnosis of health disorders) and in research settings. For every input sample, 10,000 up to 1,000,000 cells are measured, each described by 10 up to 20 features. However, recent technological advances in flow cytometry result in techniques that will be able to measure up to 100 dimensions. Standard practice is that pathologists

or researchers manually detect different cell populations in the sample by iteratively identifying regions of interest in two-dimensional scatter plots, a process that is labor-intensive and subjective.

One of the first steps in the automatic analysis of this type of data is the use of clustering techniques to identify populations of cells in an objective and reproducible way. Important challenges for clustering these datasets include: populations with different densities; populations that are not elliptic-shaped; rare populations that can be easily confused with noise; hierarchical populations that consist of several sub-types;... Another challenge is that cells can be in transition from one population to another, and thus can belong to several clusters at the same time. As individual patient samples may consist of tens of thousands up to millions of cells, and clinical datasets often consist of hundreds of patient samples, scalability is an important aspect as well. For all these reasons, a number of dedicated clustering techniques for flow cytometry data have been developed [2, 7, 19, 23].

Another important task in the automatic analysis of flow cytometry data is classifying the samples into different groups. This can serve several medical purposes: are they healthy or sick; which variant of the disease do they have; does the medicine work or not? In research settings, the goal often is to learn about which features or cell types are important to differentiate between the groups. Manually, properties of the clusters, such as the relative cell count from a certain cell type, are used to analyze the data. By using automatic data mining techniques, we can make this process easier and more objective. One possible approach is to use propositionalization, based on the cell clusters that have been manually or automatically detected. This requires that the clusters of the different patients are *aligned*, i.e. corresponding clusters need to be identified. This can be done by performing one global clustering, taking the cells of all patients together, and then for each patient indicate the subset of cells belonging to each obtained cluster. However, as cell populations may be shifted from one patient to another, global clustering is likely to perform poorly. Another approach is to perform the clustering individually per patient and then try to match the clusters. However, it is known that certain diseases are characterised by lacking some cell populations, which results in a different number of clusters per patient. These issues have motivated us to explore the use of ILP techniques in this context.

4 Complex aggregates over clusters

In this section, we assume a primary table containing the training examples, that has a one-to-many relationship with a secondary table. For a given training example, we want to aggregate over clusters of its related objects in the secondary table. In the flow cytometry use case, the primary table contains the patients, and the secondary table the measured cell properties. Each patient has ten thousands of cells or more in the secondary table. The clusters represent the different cell types present in the patient's sample. When aggregating over clusters instead of

over a subset defined by conditions on attributes, there are a number of questions to be addressed.

First, do we search for a similar cluster structure for each example, or do we cluster the objects related to each example independently? The former approach corresponds to propositionalization, and as explained in the previous section, can be performed for instance by clustering over the complete secondary table. The latter approach requires relational techniques and has the advantage that it is more flexible: the number of clusters can be optimised per example, and it can deal with clusters that are shifted among examples. In this paper we have chosen the second option, and the rest of this discussion also assumes this option.

Second, which type of clustering do we want? One can distinguish total versus partial clustering, overlapping versus non-overlapping clustering, partitional versus hierarchical clustering,... Partitional (flat) clustering algorithms return a flat, unstructured, set of clusters. Hierarchical clustering algorithms, on the other hand, return a cluster hierarchy, the top element corresponding to the complete set of related secondary objects, and the bottom elements to singleton sets.

Third, how do we represent the clustering, and the corresponding complex aggregates? Here, we present two approaches, distinguished by when the clustering step is performed. In general, we would like to express aggregates like "the average value of fluorescent marker M of the cells in a particular cluster is larger than 3", or, in other words, "there exists a cluster of cells with average value for marker M larger than 3". Thus, we obtain the following notation for complex aggregates over clusters: $\exists C, F(V, C, R)$, where C denotes a cluster, and the other variables are as above.

The first approach that we discuss is the case where the clustering step is done in pre-processing. Thus, the clustering is done before and independently of the prediction step. This allows to pre-compute aggregate functions on the obtained clusters. This way, the original data objects of the secondary table can be discarded, retaining only the (aggregated) discovered clusters. In the case of partitional clustering, this highly reduces the cardinality of the relation, resulting in a more efficient prediction step. In the case of hierarchical clustering, this gain depends on the depth of the cluster hierarchy: if the cluster hierarchy is constructed until singleton leaf nodes, the cardinality will be increased instead. One solution is to apply a top-down clustering procedure [4] and use a stopping criterion. A possible disadvantage of this approach is that if there are many examples and many related objects, the clustering step (which is performed for each example independently) may be very time-consuming, while in the final model, only few of the clusterings may actually be used.

The second approach is particularly suited for hierarchical clustering. In this case, a more flexible and efficient refinement strategy is possible, by iteratively performing a clustering step, as illustrated by the following example. Suppose we have a rule stating that a patient has a disease if the sum of the values for marker M of his cells is larger than 50. According to the theory developed in [22], this rule can be specialised in three ways: by decreasing the aggregate function (e.g., replace sum by average or maximum), by increasing the threshold value

(e.g., change 50 to 100), or by reducing the set to aggregate on. The latter can be achieved by performing a splitting step on the cells for the patients for which the original rule holds, resulting in a rule that states that there exists a cluster of cells for which the sum of the marker M values is higher than 50. Similarly, a rule stating that there is a cluster of cells with maximum value for marker M smaller than 5 can be specialised a.o. by enlarging the set to aggregate on, i.e., by enlarging the cluster. This can be accomplished by performing a merging step on the clusters for which the original rule holds. This approach leads to a more efficient clustering process than the precomputed (hierarchical) clustering approach above, in the sense that a clustering step (splitting or merging) is only performed "on demand", on the relevant examples, and on the relevant clusters of these examples. On the other hand it requires more memory space than the previous approach, as the original data objects need to be retained.

In this paper we take a first step towards complex aggregates over clusters by exploring the first approach, i.e., using a clustering computed in a pre-processing step.

5 Experiments

In this section, we first describe experiments on two synthetic datasets to show the strengths of aggregating over clusters. Afterwards, we show promising results on a real-world flow cytometry dataset. In all experiments we use a flat clustering approach. We use the relational decision tree learner Tilde[3] to learn the predictive model. We use default parameters for Tilde, except for pruning, which is turned off, and the language bias, which is discussed in detail for each dataset. Thresholds are obtained by discretisation (into ten bins).

5.1 Synthetic dataset 1

We start the experiments with a simple dataset of 20 samples, each containing 5000 cells and belonging to one of two classes. For illustration purposes, these samples only contain one cluster each. One such dataset is presented in Figure 1; we generated 6 replicates in total, and report average results.

In many cases, prior background knowledge about the structure of the data is available. In this example, we assume that we know that the clusters can be approximated by ellipses. As such, we fit an ellipse to the data, and use the mean value, the length of the major and minor axis and the angle of the major axis as features to describe the cluster. Note that using the mean value of each dimension, or using the number of cells in the cluster would not discriminate between the two classes.

We have compared four different aggregation levels: checking the existence of individual cells with one of the dimensions smaller or larger than some threshold (*no aggregates*); comparing the maximum, minimum, or average of each dimension to a threshold (*simple aggregates*); comparing the count, maximum, minimum, or average of cells with a condition on their dimensions to a threshold

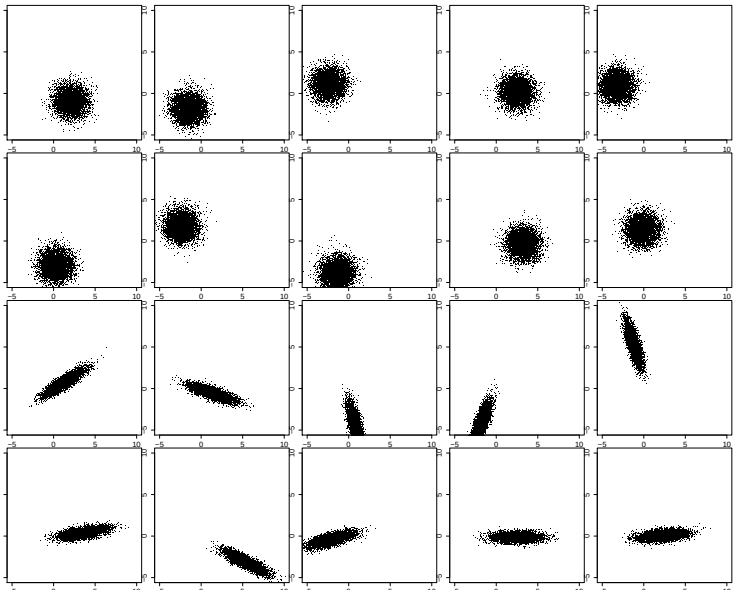

Fig. 1. Synthetic dataset 1.

(*complex aggregates*); and checking the existence of a cluster with a condition on the number (or percentage) of cells or on the mean, axis lengths or axis angle of the fitted ellipse (*cluster aggregates*).

We compare the different approaches by using them as language bias in the relational decision tree learner TILDE. We use five-fold cross-validation, predicting four patients in each fold. Figure 2 compares accuracy values of the different settings.

We see that the cluster aggregates clearly outperform the other settings. Moreover, the simple and complex aggregates perform worse than selection of individual cells. Inspection of the size of the trees (data not included) shows that the latter yields the largest trees, checking for the existence of cells in different parts of the space. The average number of internal nodes for the different settings in the figure is 3.5, 3.4, 2.6, and 1, respectively. The cluster aggregates are able to classify the data (almost) perfectly with a single test that checks the length of the major or minor axis.

5.2 Synthetic dataset 2

This dataset reflects the properties of flow cytometry data more closely. It again consists of twenty samples, which might e.g. represent patients, that must be classified into two classes, e.g. healthy or diseased.

For each sample, a number of datapoints in a four dimensional space is generated. The datapoints are divided into five clusters, which each have normal

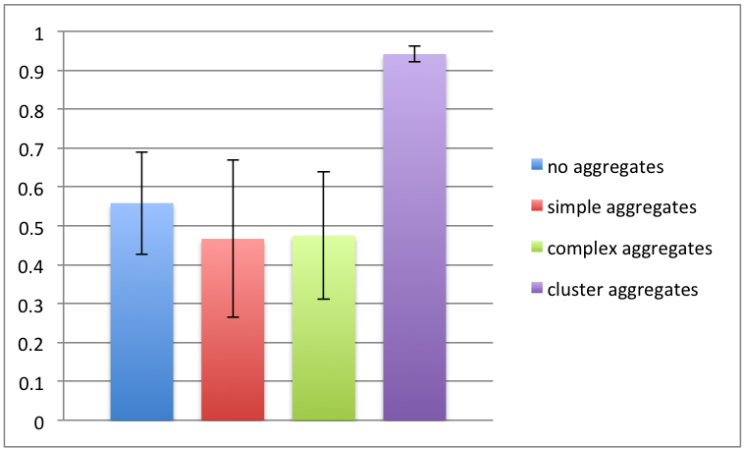

Fig. 2. Accuracy of relational decision tree with different language biases on synthetic dataset 1.

Table 1. Properties of synthetic dataset 2.

	Dim 1		Dim 2		Dim 3		Dim 4		Count	Count
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	class 1	class 2
Cluster 1	2	1.0	2	1.0	4	0.5	2	1.0	5000	5000
Cluster 2	1	0.5	1	0.5	1	0.5	2	1.0	1000 or 2000	1000 or 2000
Cluster 3	1	0.5	3	0.5	1	0.5	2	1.0	1000	1000
Cluster 4	3	0.5	1	0.5	1	0.5	2	1.0	2000 or 1000	2000 or 1000
Cluster 5	3	0.5	3	0.5	1	0.5	2	1.0	**200**	**500**

distributions. For each patient, the properties of these clusters are slightly different, by perturbing the values in Table 5.2 using a normal distribution with a small standard deviation around the given values.

As can be seen in the table, cluster five is the important cluster in the diagnosis. Depending on the abundance of datapoints in this cluster, the patients can be split in the two classes. This cluster can be defined as having high values in dimension 1 and 2, low values in dimension 3 and neutral values in dimension 4. Cluster one and four are always present, while clusters two and three can be present in different abundances, without influencing the diagnosis. An illustration of this dataset can be seen in Figure 3, in total 6 replicates were used, and average results are reported.

To test our algorithmic approach, we again built four different versions of the language bias as input to Tilde. The settings *no aggregates*, *simple aggregates*, and *complex aggregates* are the same as with the previous dataset, except that there are four dimensions now. For the *cluster aggregates*, we assume we have

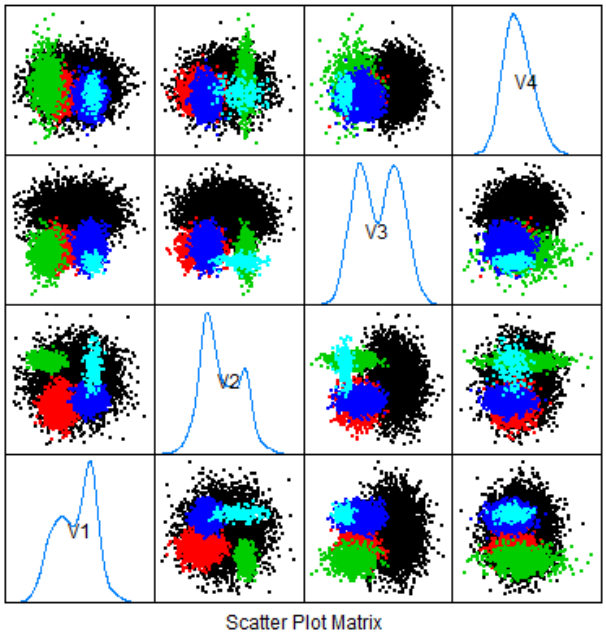

Fig. 3. One sample from synthetic dataset 2. The four-dimensional sample is visualised in all possible two-dimensional plots. The colours indicate the different clusters.

a good clustering algorithm suited for the data. Because we do not want our results to be influenced by the choice of the clustering algorithm, we use the correct clusters, known from generating the data. We describe each cluster by its number of cells and its average value of each dimension.

Accuracy values for five-fold cross-validation can be found in Figure 4. We see that the simple aggregates and the cluster aggregates perform best. The latter has a large standard deviation, because on one of the replicates, it only obtains an accuracy of 0.3. The complex aggregates yield the smallest trees, but this seems to harm their predictive accuracy.

5.3 HVTN

The HVTN (HIV Vaccine Trials Network) dataset was one of the datasets analysed in the FlowCAP II Challenge [1]. This challenge can be seen as a benchmark tool for sample classification based on flow cytometry data. The study contained 48 individuals, each of them having received an experimental HIV vaccine. Samples were collected approximately 10 months later and T-cells were challenged with two antigens. The response of T-cells was measured by flow cytometry for each of these groups, resulting in two sample files per individuals. The goal of the challenge is to discriminate between the two antigen stimulation groups.

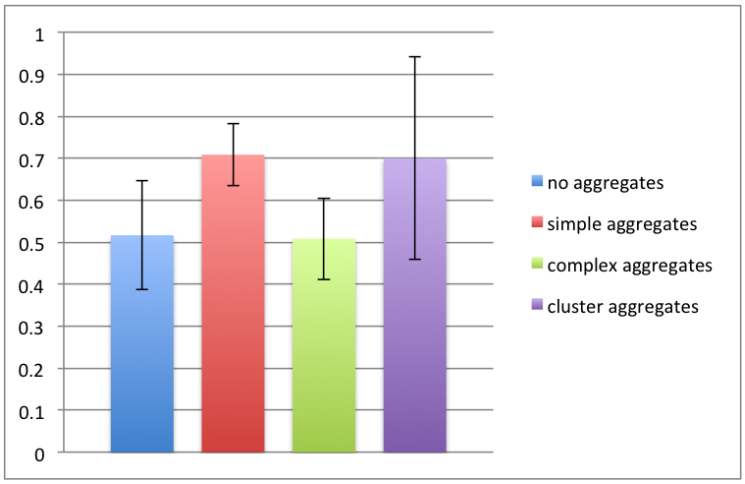

Fig. 4. Accuracy of relational decision tree with different language biases on synthetic dataset 2.

Aghaeepour et al. [1] report modest results of previous manual analysis, while several classification algorithms that participated in the challenge obtained an (almost) perfect classification, with accuracy ranging from 0.81 to 1.00.

The sample files from 27 individuals were included in the training set, while the files from the other 21 individuals were used as test set. The number of cells that is measured in each file ranges from 40,029 to 354,084.

Experimental set-up. We have kept the same train/test separation as in the challenge. On each of the sample files we performed the FlowMeans [2] algorithm, which can be seen as an adaptation of k-Means to flow cytometry data. Ten markers (attributes) were used to perform the clustering, the two markers that were left out are known to be unrelated to cell populations. The number of clusters was optimised per sample file, which resulted in 3 to 9 clusters.

For each resulting cluster, the features that we included in Tilde are the absolute and relative cell count, and the mean and standard deviation of each of the ten markers. Each feature was compared to 10 threshold values, obtained by discretisation. Furthermore, we added a complex aggregate that counts the number of clusters with one of these features. Finally, for each of the ten markers, we also added some statistics over the complete sample (i.e., without clustering): mean, standard deviation, skewness, kurtosis, median, and interquartile range. With these features, a bagging ensemble of 100 Tilde trees was run.

An schematic overview of this set-up can be seen in Figure 5.

Results. Using the procedure described above, we obtained a test set accuracy of 0.93. We are unaware whether the systems that participated in the challenge

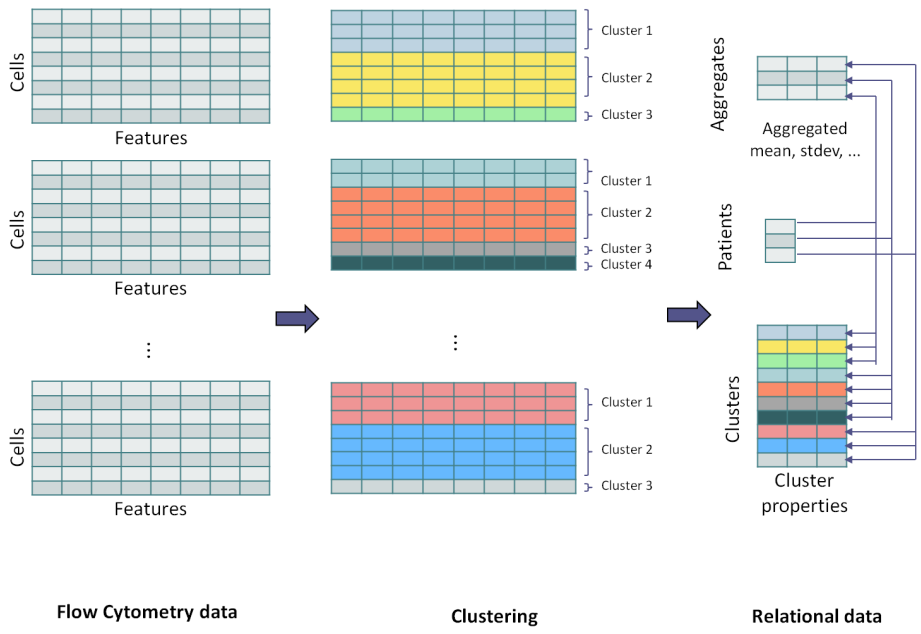

Fig. 5. Schematic overview of the experimental set-up.

took into account the fact that for each individual there is one file for each antigen stimulation. Since it is known which files belong to which individual, we also evaluated our results using this extra information. For each individual, we checked which prediction was made with the most certainty, and then predicted the opposite class for the other prediction. This resulted in an improved test set accuracy of 0.95. Still, six out of nine systems that participated in the challenge have a better predictive performance, which motivates us to look into better clustering procedures, or better ways to extract features from the predicted clusters, in future work.

We also ran a random forest instead of bagging, using feature sampling at each node of the tree. However, this led to severe overfitting of the training data, which we are also still looking into.

6 Conclusions and further work

Flow cytometry data are relationally structured, involving a one-to-many relation with very high cardinality. Handling this relation by selecting specific elements is too fine grained, while aggregating over the set is too coarse grained. This domain clearly needs complex aggregates, where the aggregates are defined over different cell populations. As it is difficult to express such populations in terms of attribute conditions, we here propose to aggregate over clusters. To our knowledge, this has not been considered before. In this paper, we proposed

two approaches to aggregate over clusters: either perform a clustering in a preprocessing step, which allows to discard the original data objects, or integrate a (hierarchical) clustering within the prediction step, which allows an efficient refinement strategy. We empirically demonstrated the potential of the first approach on flow cytometry data, while at the same time showing that there is still room for improvement. This work is the first attempt at analysing flow cytometry data with relational learning techniques, until now no clustering was considered for each example independently; further work will include developing better clustering strategies and better ways to extract features from these clusters.

Acknowledgments

Celine Vens is a Postdoctoral Fellow of the Research Foundation - Flanders (FWO). Sofie Van Gassen is funded by a Ph.D. grant of the Agency for Innovation by Science and Technology (IWT).

References

1. Aghaeepour, N., Finak, G., Consortium, F., Consortium, D., Hoos, H., Mosmann, T., Brinkman, R., Gottardo, R., Scheuermann, R.: Critical assessment of automated flow cytometry data analysis techniques. Nat. Methods 10(3), 228–238 (2013)
2. Aghaeepour, N., Nikolic, R., Hoos, H.H., Brinkman, R.R.: Rapid cell population identification in flow cytometry data. Cytometry Part A 79(1), 6–13 (2011)
3. Blockeel, H., De Raedt, L.: Top-down induction of first order logical decision trees. Artificial Intelligence 101(1-2), 285–297 (June 1998)
4. Blockeel, H., De Raedt, L., Ramon, J.: Top-down induction of clustering trees. In: Proc. of the 15th International Conference on Machine Learning,. pp. 55–63 (1998)
5. Blockeel, H., Bruynooghe, M.: Aggregation versus selection bias, and relational neural networks. In: IJCAI-2003 Workshop on Learning Statistical Models from Relational Data, SRL-2003 (2003)
6. Charnay, C., Lachiche, N., Braud, A.: Incremental construction of complex aggregates: Counting over a secondary table. In: Online preprints of 23th International Conference on Inductive Logic Programming. pp. 1–6 (2013)
7. Finak, G., Bashashati, A., Brinkman, R., Gottardo, R.: Merging mixture components for cell population identification in flow cytometry. Advances in Bioinformatics 2009 (2009)
8. Frank, R., Moser, F., Ester, M.: A method for multi-relational classification using single and multi-feature aggregation functions. In: Proc. of the 11th European Conference on Principles of Data Mining and Knowledge Discovery. Lecture Notes in Computer Science, vol. 4702, pp. 430–437. Springer (2007)
9. Frasconi, P., Jaeger, M., Passerini, A.: Feature discovery with type extension trees. In: Proceedings of 18th International Conference on Inductive Logic Programming. Lecture Notes in Computer Science, vol. 5194, pp. 122–139. Springer (2008)
10. Herzenberg, L., Tung, J., Moore, W., Herzenberg, L., Parks, D.: Interpreting flow cytometry data: a guide for the perplexed. Nature Immunology 7(7), 681–685 (2006)

11. Jaeger, M., Lippi, M., Passerini, A., Frasconi, P.: Type extension trees for feature construction and learning in relational domains. Artificial Intelligence 204, 30?55 (November 2013)
12. Knobbe, A., Siebes, A., Marseille, B.: Involving aggregate functions in multi-relational search. In: Proc. of the 6th European Conference on Principles of Data Mining and Knowledge Discovery. pp. 287–298. Springer Verlag (August 2002)
13. Koller, D.: Probabilistic relational models. In: Proceedings of the 9th International Workshop on Inductive Logic Programming. Lecture Notes in Artificial Intelligence, vol. 1634, pp. 3–13. Springer Verlag (1999)
14. Krogel, M.A., Wrobel, S.: Facets of aggregation approaches to propositionalization. In: Horváth, T., Yamamoto, A. (eds.) Proceedings of the Work-in-Progress Track at the 13th International Conference on Inductive Logic Programming. pp. 30–39 (2003)
15. Muggleton, S. (ed.): Inductive Logic Programming. Academic Press (1992)
16. Neville, J., Jensen, D., Friedland, L., Hay, M.: Learning relational probability trees. In: Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 625–630. ACM Press (2003)
17. Perlich, C., Provost, F.: Aggregation-based feature invention and relational concept classes. In: Proceedings of the ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 167–176. ACM Press (2003)
18. Srinivasan, A., Muggleton, S., King, R.: Comparing the use of background knowledge by inductive logic programming systems. In: De Raedt, L. (ed.) Proceedings of the 5th International Workshop on Inductive Logic Programming. pp. 199–230 (1995)
19. Sugár, I.P., Sealfon, S.C.: Misty mountain clustering: application to fast unsupervised flow cytometry gating. BMC bioinformatics 11(1), 502 (2010)
20. Uwents, W., Blockeel, H.: Classifying relational data with neural networks. In: Proceedings of 15th International Conference on Inductive Logic Programming, Bonn, Germany. Lecture Notes in Artificial Intelligence, vol. 3625, pp. 384–396. Springer (2005)
21. Van Assche, A., Vens, C., Blockeel, H., Džeroski, S.: First order random forests: Learning relational classifiers with complex aggregates. Machine Learning 64(1-3), 149–182 (2006)
22. Vens, C., Ramon, J., Blockeel, H.: Refining aggregate conditions in relational learning. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) Proc. of the 10th European Conference on Principles of Data Mining and Knowledge Discovery. Lecture Notes in Artificial Intelligence, vol. 4213, pp. 383–394. Springer (September 2006)
23. Zare, H., Shooshtari, P., Gupta, A., Brinkman, R.R.: Data reduction for spectral clustering to analyze high throughput flow cytometry data. BMC bioinformatics 11(1), 403 (2010)